# Intrusion Detection Using Conditional Random Fields

Siddheshwar V. Patil[1], Prakash J. Kulkarni[2]
[1] Department of Computer Science and Engineering, Walchand College of Engineering, Sangli, MH, India
Email: Siddheshwar.patil@gmail.com
[2] Department of Computer Science and Engineering, Walchand College of Engineering, Sangli, MH, India
Email: pjk_walchand@rediffmail.com

***Abstract* - Intrusion detection systems have become a key component in ensuring the safety of systems and networks. This paper introduces the probabilistic approach called Conditional Random Fields (CRF) for detecting network based intrusions. In this paper, we have shown results for the issue of accuracy using CRFs. It is demonstrated that high attack detection accuracy can be achieved by using Conditional Random Fields. Experimental results on the benchmark KDD'99 intrusion data set show the importance of proposed system. The improvement in attack detection accuracy is very high for Probe, Denial of Service, U2R and R2L attacks.**

***Index Terms* – Intrusion Detection, Conditional Random Fields, Network Security**

## I. Introduction

With the growth of computer networks usage and the huge increase in the number of applications running on top of it, network security becomes an important area of research and study. Intrusion detection is becoming enormously popular, which is at the core of network security. The role of Intrusion Detection Systems (IDSs) is to detect anomalies and attacks in the network. The work in the intrusion detection field has been mostly focused on anomaly-based and misuse-based detection techniques for a long time. The former is generally used in commercial products while the anomaly-based detection is favoured in studies and research. Anderson's work in 1980s was the entry point for most of the aspirants working in the domain of Intrusion Detection [1]. Intrusion Detection Systems (IDSs) are special-purpose devices to detect anomalies and attacks in the network. They are categorized into network based systems, host based systems, or application based systems depending on deployment mode and data available for analysis [2]. Intrusion detection systems can also be classified as signature based systems or anomaly based systems depending on the detection methods for attacks. The signature-based systems are trained by extracting signatures from known attacks while the anomaly-based systems learn from analysis of network behaviors where the normal data collected when there is no anomalous activity. Hybrid approach is one that combines advantages of both, signature based systems and anomaly based systems. The objective of this paper is to develop a system that can detect most of the network attacks with maximum accuracy. It also ensures that the desired system will handle large amount of network traffic and it will have better decision making. This paper is organized as follows: In Section 2, we describe the use of Conditional Random Fields (CRFs) [3] for intrusion detection. Experimental work is given in Section 3. The results shows that the proposed system using CRFs performs better, reducing the false alarm rate.

## II. CONDITIONAL RANDOM FIELDS

The Conditional Random Field is probabilistic mathematical model [4]. For better understanding of this model, let us have following explanation. Let 'X' be the random variable over data sequence to be labeled and 'Y' the corresponding label sequence. Let G = (V, E) be a graph, such that $Y = (Y_\upsilon)\ \upsilon \in (v)$ so that 'Y' is indexed by the vertices of 'G'. Then, (X, Y) is a CRF, when conditioned on X, the random variables $Y_\upsilon$ posses the Markov property with respect to the graph

$$p\,(Y_\upsilon \mid X,\ Y_{w,}\ w \neq \upsilon) = p\,(Y_\upsilon \mid X,\ Y_{w,}\ w \approx \upsilon)$$

where w H" υ means that 'w' and 'υ' are neighbors in 'G', i.e., a CRF is a random field globally conditioned on 'X'. For a simple sequence (or chain) modeling, as in our case, the joint distribution over the label sequence 'Y' given 'X' has the following form:

$$p(y|x) \propto \exp\left(\sum_{e\in E,k} \lambda k\ fk\,(e, y|e, x) + \sum_{\theta\in V,k} \mu k\ gk(\vartheta, y|\vartheta, x)\right)$$

where 'x' is the data sequence, 'y' is a label sequence, and y|s is the set of components of 'y' associated with the vertices or edges in subgraph 'S'. In addition, the features $f_k$ and $g_k$ are assumed to be given and fixed. For example, a boolean edge feature $f_k$ might be true, if the observation $X_i$ is 'protocol=tcp', tag $Y_{i-1}$ is 'normal' and tag $Y_i$ is 'normal' [5] [6]. Further, the parameter estimation problem is to find the parameters
θ= (λ1, λ2, λ3... μ1, μ2, μ3.....) from the training data.
The model showed in figure 1 can also be explained in other way. As an example, when we classify the network connections as either normal or as attack, a system may consider features such as 'logged in' and 'number of file creations'. Individual analysis of these features does not provide any meaningful information for detecting the attacks. However, when these features are analyzed together, the information obtained is found to be helpful for the classification task. This information becomes more concrete and aids in classification when

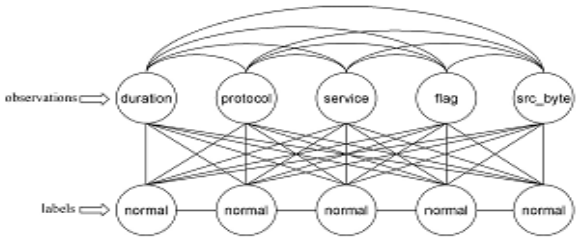


Fig. 1. CRF Graphical Representation




DOI: 01.IJNS.02.03.125

analyzed with other features such as 'protocol type' and 'amount of data transferred' between source and destination. These relationships, between different features in the observed data, if considered during classification can significantly decrease classification error. The CRFs do not consider features to be independent and hence perform better. The graphical representation of CRF is shown in figure 1 .

## III. EXPERIMENTAL WORK

The benchmarked KDD'99 intrusion data set is used for experimentation [7]. In the experiments, 10 percent of the total training data, 10 percent of the test data (with corrected labels) and old test data (labeled) is used. This leads to 494,020 training instances, 311,029 test instances from corrected test data and 22,544 test instances from old test data. A connection is a sequence of TCP packets starting and ending at some well defined times, in which, data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes. Further, every record is represented by 41 different features. Each record represents a separate connection and is hence considered to be independent of any other record. The training data is either labeled as normal or as one of the 24 different kinds of attack. These 24 attacks can be grouped into four classes: Probe, DoS, R2L, and U2R. Similarly, the test data is also labeled as either 'normal' or as 'attack', belonging to the four attack groups. The test data is not from the same probability distribution as the training data, and it includes specific attack types not present in the training data. This makes the intrusion detection task more realistic. To check the accuracy of the system, the terms like Precision, Recall and F-Value are considered. These can be defined as follows:

$$Precision = \frac{TP}{(TP + FP)}$$

$$Recall = \frac{TP}{(TP + FN)}$$

$$F - Value = \frac{(1 + \beta^2) \times Recall \times Precision}{\beta^2 \times (Recall + Precision)}$$

where TP, FP and FN are the number of True Positives, False Positives and False Negatives, respectively and β corresponds to the relative importance of precision versus recall and is usually set to 1.

### A. Probe Attack

The probe attacks are aimed at acquiring information about the target network from a source that is often external to the network. Hence, basic connection level features such as the 'duration of connection' and 'source bytes' are significant while features like 'number of files creations' and 'number of files accessed' are not expected to provide information for detecting probes. For probe attack, there are 5 significant features out of 41 features. So, this feature selection is done manually by having the domain knowledge by the experts. Table I shows the features selected for probe class. After selecting these 5 features, we have formed the probe patterns by using CRF coding in Java programming language [8]. For this purpose, we used the records from 10 percent KDD train data which is of type 'Normal + Probe'. For example, to detect probe attacks, we train and test the system with probe and normal data only. So, the total 101,385 records are available from kdd train dataset. We do not add the DOS, R2L and U2R data when detecting Probes. This allows the system to better learn the features for probe and normal events. After that, we tested it with two labeled datasets, 10 percent corrected KDD test data and old test data. Figure 2 shows probe class representation while Table II shows the results for probe attack.


Fig. 2. Probe Class Representation

TABLE I
FEATURE SELECTED FOR PROBE CLASS

| Feature Number | Feature Name |
|---|---|
| 1 | duration |
| 2 | protocol_type |
| 3 | service |
| 4 | flag |
| 5 | src_bytes |

TABLE II
RESULTS FOR PROBE ATTACK

| PROBE DETECTION | PRECISION | RECALL | F-VALUE |
|---|---|---|---|
| CORRECTED KDD TEST DATA | 87.09 | 97.87 | 92.17 |
| OLD KDD TEST DATA | 85.79 | 97.73 | 91.37 |

### B. DOS Attack

For the DoS attack, traffic features such as the 'percentage of connections having same destination host and same service' and packet level features such as the 'source bytes' and 'percentage of packets with errors' are significant. To detect DoS attacks, it may not be important to know whether a user is 'logged in or not'. For DOS attack, there are 9 significant features out of 41 features. Table III shows the features selected for DOS class. After selecting these 9 features, we have formed the DOS patterns by using CRF coding in Java programming language. For this purpose, we used the records from 10 percent KDD train data which is of type 'Normal + DOS'. For example, to detect DOS attacks, we train and test the system with DOS and normal data only. So, the total 488,736 records are available from KDD train dataset.




DOI: 01.IJNS.02.03.125

We do not add the probe, R2L and U2R data when detecting DOS. This allows the system to better learn the features for DOS and normal events. After that, we tested it with 10 percent corrected KDD test data and old test data. Table IV shows the results for DOS attack.

### *C. R2L Attack*

The R2L attacks are one of the most difficult to detect as they involve the network level and the host level features. So, both the network level features such as the 'duration of connection' and 'service requested' and the host level features such as the 'number of failed login attempts' among others are important for detecting R2L attacks. For R2L attack, there are 14 significant features out of 41 features. Table V shows the features selected for R2L class. After selecting these 14 features, We have formed the R2L patterns by using CRF coding in Java programming language. For this purpose, we used the records from 10 percent KDD train data which is of type 'Normal + R2L'. For example, to detect R2L attacks, we train and test the system with R2L and normal data only. So, the total 98,404 records are available from kdd train dataset. We do not add the Probe, DoS and U2R data when detecting R2L. This allows the system to better learn the features for R2L and normal events. After that, we tested it with 10 percent corrected KDD test data and old test data. Table VI shows the results for R2L attack .

TABLE III
FEATURE SELECTED FOR DOS CLASS

| Feature Number | Feature Name |
|---|---|
| 1 | duration |
| 2 | protocol_type |
| 4 | flag |
| 5 | src_bytes |
| 23 | count |
| 34 | dst_host_same_srv_rate |
| 38 | dst_host_serror_rate |
| 39 | dst_host_srv_serror_rate |
| 40 | dst_host_rerror_rate |

TABLE IV
RESULTS FOR DOS ATTACK

| DOS Detection | Precision | Recall | F-Value |
|---|---|---|---|
| Corrected KDD Test Data | 99.72 | 90.92 | 95.12 |
| Old KDD Test Data | 98.88 | 87.68 | 92.94 |

### *D. U2R Attack*

The U2R attacks involve the semantic details that are very difficult to capture at an early stage. Such attacks are often content based and target an application. Hence, for U2R attacks, we selected features such as 'number of file creations' and 'number of shell prompts invoked', while we ignored features such as 'protocol' and 'source bytes'. For U2R attack, there are 8 significant features out of 41 features. Table VII shows the features selected for U2R class.

TABLE V
FEATURE SELECTED FOR R2L CLASS

| Feature Number | Feature Name |
|---|---|
| 1 | duration |
| 2 | protocol_type |
| 3 | service |
| 4 | flag |
| 5 | src_bytes |
| 10 | hot |
| 11 | num_failed_logins |
| 12 | logged_in |
| 13 | num_compromised |
| 17 | num_file_creations |
| 18 | num_shells |
| 19 | num_access_files |
| 21 | is_host_login |
| 22 | is_guest_login |

TABLE VI
RESULTS FOR R2L ATTACK

| R2L Detection | Precision | Recall | F-Value |
|---|---|---|---|
| Corrected KDD Test Data | 100.00 | 36.28 | 53.25 |
| Old KDD Test Data | 100.00 | 50.00 | 66.67 |

After selecting these 8 features, we have formed the U2R patterns by using CRF coding in Java programming language. For this purpose, we used the records from 10 percent KDD train data which is of type 'Normal + U2R'. For example, to detect U2R attacks, we train and test the system with U2R and normal data only. So, the total 97,330 records are available from kdd train dataset. We do not add the Probe, DoS and R2L data when detecting U2R. This will allows the system to better learn the features for U2R and normal events. After that, we tested it with 10 percent corrected KDD test data and old test data. Table VIII shows the results for U2R attack. We used domain knowledge together with the feasibility of each feature before selecting it for a particular attack class. Thus, from the total 41 features, we have selected only 5 features for Probe class, 9 features for DoS class, 14 features for R2L class, and 8 features for U2R class.

## Conclusion

In this paper, our experimental results in Section 3 show that CRFs approach is better in improving the attack detection rate and decreasing the False Alarm Rate (FAR). The FAR must be low for any intrusion detection system. The future work will be the development of signatures for signature-based systems. The signature-based systems can be




DOI: 01.IJNS.02.03. 125

deployed at the periphery of a network to filter out attacks and leaving the detection of new unknown attacks for anomaly and hybrid systems.

TABLE VII
FEATURE SELECTED FOR U2R CLASS

| Feature Number | Feature Name |
|---|---|
| 10 | hot |
| 13 | num_compromised |
| 14 | root_shell |
| 16 | num_root |
| 17 | num_file_creations |
| 18 | num_shells |
| 19 | num_access_files |
| 21 | is_host_login |

TABLE VIII
RESULTS FOR U2R ATTACK

| U2R Detection | Precision | Recall | F-Value |
|---|---|---|---|
| Corrected KDD Test Data | 100.00 | 20.51 | 34.04 |
| Old KDD Test Data | 100.00 | 21.62 | 35.55 |

## References

[1] J.P. Anderson, "Computer Security Threat Monitoring and Surveillance", http://csrc.nist.gov/publications/history/ande80.pdf , 2010.
[2] R. Bace and P. Mell, "Intrusion Detection Systems, Computer Security Division, Information Technology Laboratory, Natl Inst. of Standards and Technology", 2001.
[3] K.Gupta and B.Nath, R.Kotagiri, "Layered approach using Conditional Random Fields for Intrusion Detection", IEEE Transaction on dependable and secure computing, 2010.
[4] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", Proc. 18th Intl Conf. Machine Learning (ICML 01), pp. 282-289, 2001.
[5] K.K.Gupta, B.Nath, and R.Kotagiri, "Network Security Framework," Intl J. Computer Science and Network Security, vol. 6, no. 7B, pp. 151-157, 2006.
[6] K.K.Gupta, B.Nath, and R.Kotagiri, "Conditional Random Fields for Intrusion Detection", Proc. 21st Intl Conf. Advanced Information Networking and Applications Workshops (AINAW 07), pp. 203-208, 2007.
[7] KDD Cup 99 Intrusion Detection Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 2010.
[8] CRF++: Yet another CRF Toolkit, http://crfpp.sourceforge.net/ , 2010.




DOI: 01.IJNS.02.03.125